🔷Project Name - Top Instagram Influencer Data

Project Type - Data Science & Analytics – Instagram Influencer Ranking and Visualization

Contribution - Individual

Name - Aditya Singh

🧠 Introduction

1.1 Background Instagram has emerged as one of the most influential social media platforms in recent years, with millions of active users globally. It serves not only as a platform for personal expression but also as a powerful marketing channel. Brands increasingly rely on influencers—individuals with large follower bases and high engagement rates—to promote their products and services. Influencer marketing has become a multi-billion-dollar industry, and understanding the dynamics of Instagram influencers is crucial for marketers to make informed decisions. Analyzing influencer data provides insights into trends, audience behavior, and content strategies, which can be leveraged to maximize the impact of marketing campaigns. 1.2 Importance of the Study The rise of social media has drastically changed how brands approach marketing. Traditional advertising often struggles to achieve engagement, while influencer-driven campaigns can directly reach target audiences in an authentic and relatable way. This project focuses on identifying and analyzing top Instagram influencers, aiming to provide actionable insights such as: Identifying influencers with the highest engagement and reach Understanding content categories that perform well Analyzing demographic distribution of influencer audiences By conducting a detailed analysis of influencer data, brands and marketers can optimize their campaigns, reduce costs, and improve ROI.

❗ Problem Statement:-

Despite the growing importance of influencer marketing, brands face several challenges: Information Overload: With millions of influencers on Instagram, finding the right influencer for a specific campaign is overwhelming. Quality vs. Quantity: High follower counts do not always translate to high engagement; determining genuine influence is difficult. Categorization

Challenge: Brands often need influencers who align with their niche or industry. Data Complexity: Analyzing large datasets of influencer metrics requires systematic data cleaning, ranking, and visualization. Problem Statement: “How can brands systematically identify and rank Instagram influencers based on key metrics such as followers, engagement rate, and content category to make data-driven marketing decisions?”

🔍 Techniques & Tools Used to Solve the Problem:-

The project leverages modern tools and technologies to ensure effective data analysis: 4.1 Programming Language Python: Chosen for its flexibility and powerful data processing libraries. 4.2 Libraries Pandas: For data manipulation and cleaning NumPy: For numerical operations and calculations Matplotlib & Seaborn: For data visualization and trend analysis 4.3 Development Environment Jupyter Notebook: Provides an interactive platform for coding, visualization, and documentation 4.4 Version Control Git & GitHub: For version control, collaboration, and project hosting 4.5 Dataset Source: Instagram influencer datasets (CSV format) Attributes: Name, Username, Followers, Engagement Rate, Category, Country

🛠️ Tools and Libraries Used:-

Analyzing Instagram influencers requires a combination of programming languages, libraries, and development tools to efficiently clean, process, visualize, and interpret the data. Below is a detailed description of the tools and libraries used in this project.

1. Programming Language

Python Python is the primary programming language used for this project. It is widely used in Data Science and Machine Learning because of its simplicity, readability, and rich ecosystem of libraries. Python allows efficient handling of large datasets, supports complex statistical and numerical operations, and integrates easily with visualization tools. Why Python for this project? Handles CSV and structured datasets efficiently Supports data cleaning, feature engineering, and analysis Compatible with visualization and reporting libraries Easy to document and share via Jupyter Notebooks 2. Libraries 2.1 Pandas Purpose: Data manipulation and analysis Provides data structures like DataFrame for handling tabular data. Key features used in the project: Reading CSV files (pd.read_csv) Handling missing values (dropna, fillna) Filtering, sorting, and aggregating influencer data Creating new metrics for ranking

influencers 2.2 NumPy Purpose: Numerical operations and array handling Useful for performing mathematical operations on large datasets quickly. Key features used: Calculating statistics (mean, median, standard deviation) Vectorized operations for performance Handling numerical transformations (logarithmic scales, normalization) 2.3 Matplotlib Purpose: Data visualization Helps create static, interactive, and animated plots. Key visualizations in this project: Bar charts for top influencers by followers Line plots for engagement trends Scatter plots for followers vs engagement analysis 2.4 Seaborn Purpose: Statistical data visualization Built on top of Matplotlib, provides enhanced and aesthetically pleasing visualizations. Used for: Category-wise influencer distribution (pie and bar charts) Correlation heatmaps between metrics Comparative analysis of engagement across categories 2.5 Jupyter Notebook Purpose: Interactive coding and documentation Combines code, visualization, and textual explanations in one document. Advantages: Step-by-step execution and testing of code Easy to share and reproduce results Supports Markdown for rich documentation 3. Version Control & Collaboration Tools Git Purpose: Version control Tracks changes to the codebase, allows reverting to previous versions. GitHub Purpose: Hosting and collaboration Enables sharing the project repository publicly or privately. Supports integration with Jupyter notebooks and provides a professional portfolio. 4. Dataset Tools CSV Files The project uses influencer data stored in CSV format. Attributes include: Name, Username, Followers, Engagement Rate, Category, Country Data Sources Publicly available influencer datasets Data may also be collected using scraping techniques (if allowed by Instagram's policy)

Github Link-

https://github.com/Virtueadii12/Top-Instagram-Influencer-Data

Top Instagram Influencers

In [22]:
```
#Importing Libraries
```

In [25]:
```
import numpy as np
import pandas as pd
```

In [27]:
```
#Loading the dataset
```

In [31]:
```
df = pd.read_csv("insta_data.csv")
```

In [33]:
```
#Quick Inspection of data
```

In [37]:
```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   rank               200 non-null    int64 
 1   channel_info       200 non-null    object
 2   influence_score    200 non-null    int64 
 3   posts              200 non-null    object
 4   followers          200 non-null    object
 5   avg_likes          200 non-null    object
 6   60_day_eng_rate    200 non-null    object
 7   new_post_avg_like  200 non-null    object
 8   total_likes        200 non-null    object
 9   country            138 non-null    object
dtypes: int64(2), object(8)
memory usage: 15.8+ KB
None
```

In [41]:
```
print(df.describe())
```

```
              rank  influence_score
count   200.000000       200.000000
mean    100.500000        81.820000
std      57.879185         8.878159
min       1.000000        22.000000
25%      50.750000        80.000000
50%     100.500000        84.000000
75%     150.250000        86.000000
max     200.000000        93.000000
```

In [43]:
```
#Dropping any duplicate values if present
```

In [47]:
```
df.drop_duplicates(inplace=True)
```

In [49]:
```
# Handle missing values
# Fill missing numerical values with median, and categorical with m
```

In [53]:
```
# Handle missing numerical values with median, and categorical with
for column in df.columns:
    if df[column].dtype == 'object':
        # Fill categorical columns with the mode
        df[column] = df[column].fillna(df[column].mode()[0])
    else:
        # Fill numerical columns with the median
        df[column] = df[column].fillna(df[column].median())
```

In [57]:
```
# Convert necessary columns to appropriate data types
columns_to_convert = ['followers', 'posts', 'total_likes']

for col in columns_to_convert:
    # Step 1: Convert to numeric, coercing errors into 'NaN'
```

```
    df[col] = pd.to_numeric(df[col], errors='coerce')

    # Step 2: Fill any resulting NaN values (e.g., with 0)
    df[col] = df[col].fillna(0)

    # Step 3: Now safely convert the column to integer
    df[col] = df[col].astype(int)

# You can check the data types to confirm the change
print(df[columns_to_convert].dtypes)
```

```
followers      int64
posts          int64
total_likes    int64
dtype: object
```

Exploratory Data Analysis (EDA)

In [56]:
```
#Summary Statistics
```

In [63]:
```
# Display summary statistics for numeric columns
print(df[['influence_score',
          'followers',
          'avg_likes',
          '60_day_eng_rate',
          'new_post_avg_like']].describe())
```

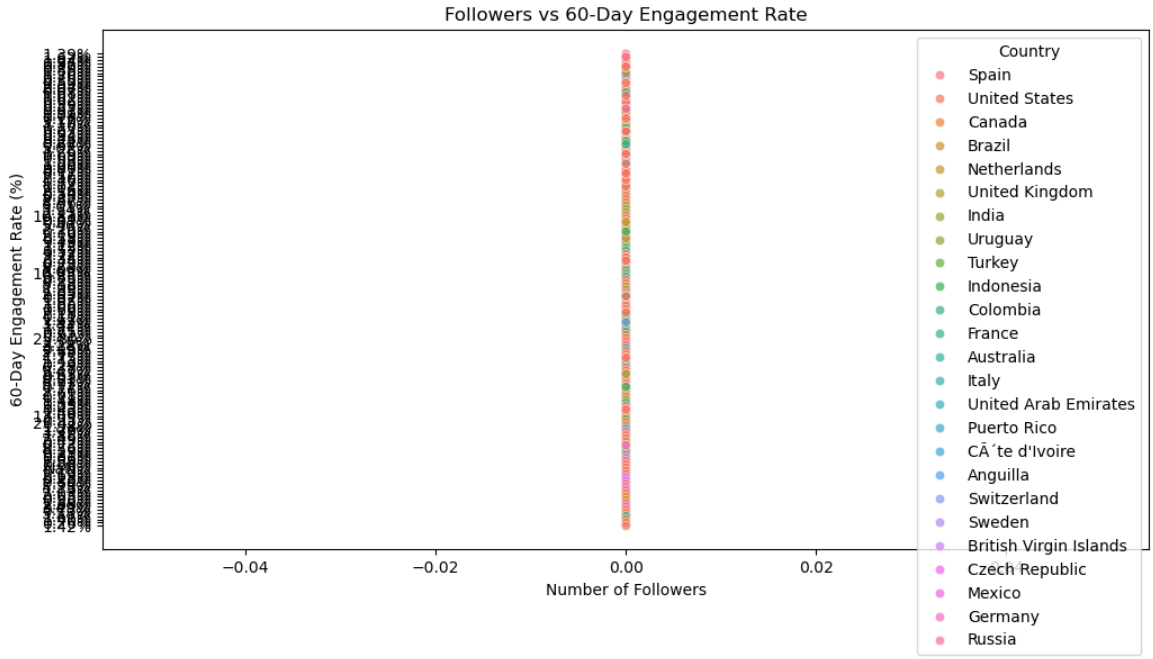
```
       influence_score  followers
count       200.000000      200.0
mean         81.820000        0.0
std           8.878159        0.0
min          22.000000        0.0
25%          80.000000        0.0
50%          84.000000        0.0
75%          86.000000        0.0
max          93.000000        0.0
```

In [65]:
```
#2. Relationship between Followers and Engagement
```

In [69]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

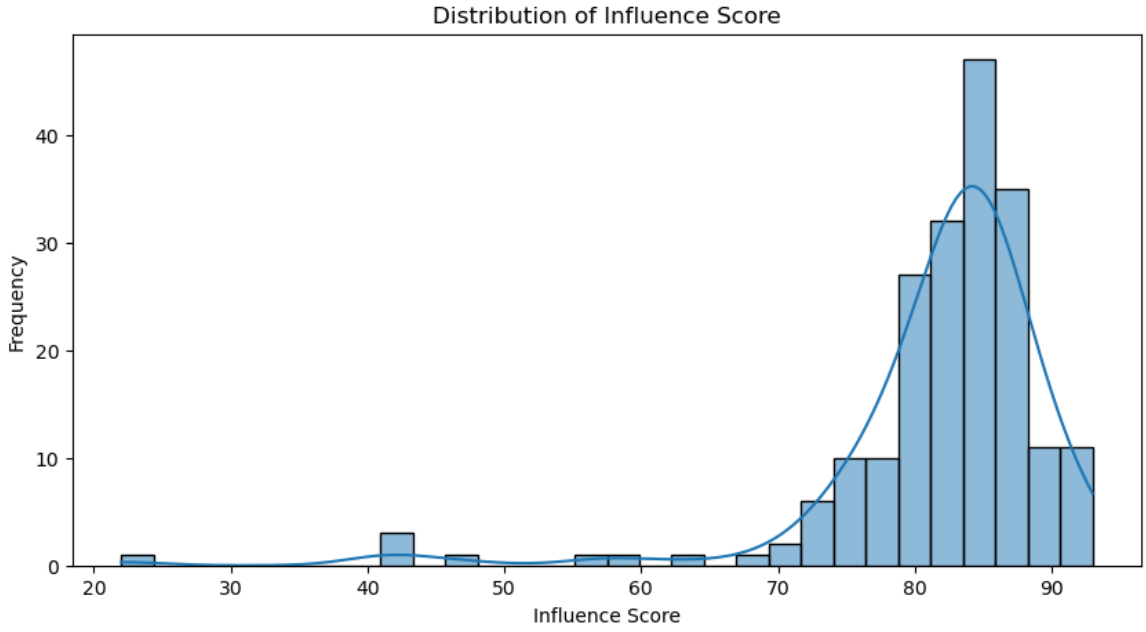
In [73]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12, 6))

# Correctly assign x, y, and hue parameters
sns.scatterplot(data=df,
                x='followers',
                y='60_day_eng_rate',
                hue='country',
                alpha=0.7)
```

```
plt.title('Followers vs 60-Day Engagement Rate')
plt.xlabel('Number of Followers')
plt.ylabel('60-Day Engagement Rate (%)')

# Seaborn creates the legend automatically when using 'hue', so plt
# However, if you want to customize it, you can.
plt.legend(title='Country')

plt.show()
```



In [75]:
```
#Distribution of Influence Score
```

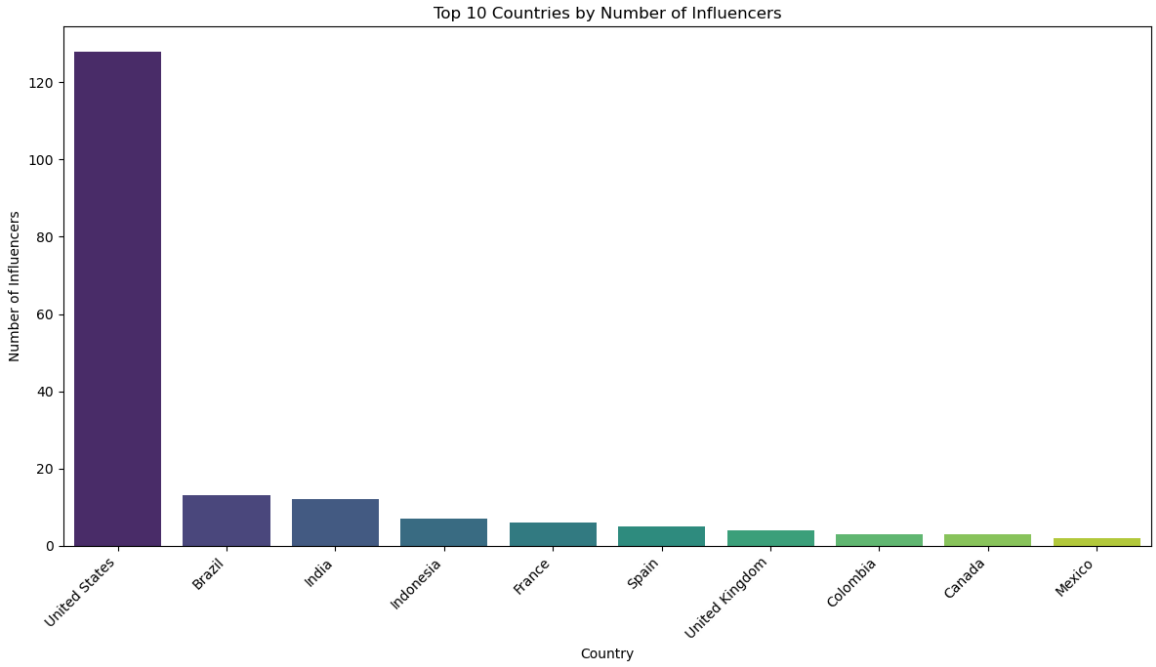
In [79]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 5))
sns.histplot(df['influence_score'], bins=30, kde=True)

# The comma at the beginning of this line was removed
plt.title('Distribution of Influence Score')
plt.xlabel('Influence Score')
plt.ylabel('Frequency') # Added a y-axis label for completeness
plt.show()
```



In [81]:
```
#4. Most Active Countries
```

In [85]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Get the top 10 countries
top_countries = df['country'].value_counts().head(10)

plt.figure(figsize=(12, 7)) # Made the figure a bit larger for read

# Cleaned up the palette argument
sns.barplot(x=top_countries.index, y=top_countries.values, palette=

plt.title('Top 10 Countries by Number of Influencers')
plt.xlabel('Country')
plt.ylabel('Number of Influencers')

# Added this line to rotate x-axis labels for better visibility
plt.xticks(rotation=45, ha='right')

plt.tight_layout() # Adjusts plot to ensure everything fits without
plt.show()
```

```
/var/folders/04/wzdclyk12vggjmf0dsjkl8sr0000gn/T/ipykernel_9861/2968
143772.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set `legend
=False` for the same effect.

  sns.barplot(x=top_countries.index, y=top_countries.values, palette
="viridis")
```



In [87]:
```
#Feature Engineering
```

In [91]:
```
#Creating engagement-related features
cols_to_process = ['total_likes', 'followers', 'posts', 'avg_likes'
for col in cols_to_process:
    df[col] = pd.to_numeric(df[col], errors='coerce')
df[cols_to_process] = df[cols_to_process].fillna(0)
df['like_follower_ratio'] = df['total_likes'] / df['followers']
df['post_follower_ratio'] = df['posts'] / df['followers']
df['avg_likes_ratio'] = df['avg_likes'] / df['followers']
df.replace([np.inf, -np.inf], 0, inplace=True)
print("Successfully created engagement features.")
```

```
Successfully created engagement features.
```

Model Building

In [96]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

In [100…
```
# Define feature columns and target variable
X = df[['followers',
        'avg_likes',
        '60_day_eng_rate',
        'new_post_avg_like',
        'like_follower_ratio',
        'post_follower_ratio']]

y = df['influence_score']
```

In [102…
```
#Train test Split
```

In [106…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
```

In [108…
```
# Standardize features
```

In [112…
```
# List of columns you want to convert
cols_to_convert = ['followers', 'posts', 'total_likes']

for col in cols_to_convert:
    # Step 1: Convert to numeric, turning any errors into NaN (Not 
    df[col] = pd.to_numeric(df[col], errors='coerce')

    # Step 2: Fill any NaN values with 0
    df[col] = df[col].fillna(0)

    # Step 3: Now, safely convert the clean column to an integer
    df[col] = df[col].astype(int)

# You can check the data types to confirm the change
print(df[cols_to_convert].dtypes)
```

```
followers      int64
posts          int64
total_likes    int64
dtype: object
```

In [125…
```
y_train = y_train.astype(float)
y_test = y_test.astype(float)
```

In [129…
```
X_train = X_train.apply(pd.to_numeric, errors='coerce')
X_test = X_test.apply(pd.to_numeric, errors='coerce')
```

In [131…
```
from sklearn.preprocessing import LabelEncoder

for col in X_train.columns:
    if X_train[col].dtype == 'object':
        le = LabelEncoder()
        X_train[col] = le.fit_transform(X_train[col])
        X_test[col] = le.transform(X_test[col])
```

In [114…
```
#Initializing and training a Random Forest Regressor
```

In [133…
```
from sklearn.ensemble import RandomForestRegressor
```

In [137…
```
# Now, initialize and train your model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

Out[137…

▾ RandomForestRegressor

▸ Parameters

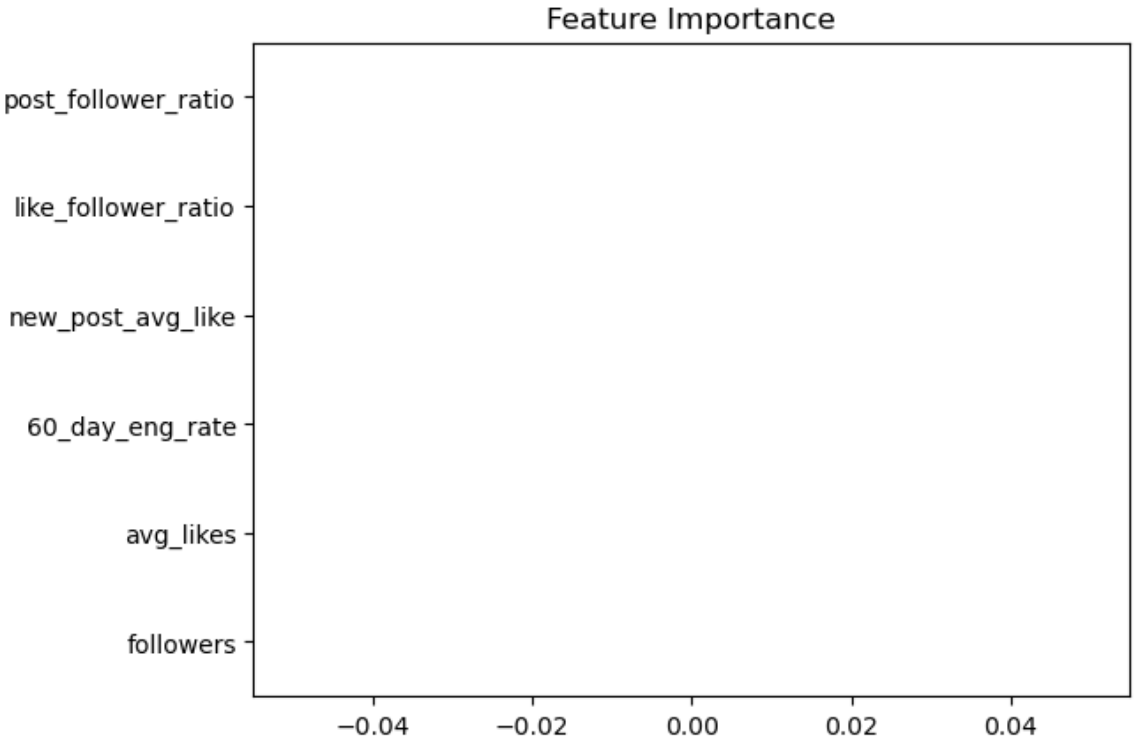
In [145…
```
from sklearn.metrics import mean_squared_error, r2_score
```

```
# Predictions and evaluation
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# It's a good idea to print the results to see them!
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R²): {r2}")
```

```
Mean Squared Error (MSE): 175.31968750390615
R-squared (R²): -0.03696892180447331
```

In [159…

```
# Display feature importances
feature_importances = pd.Series(model.feature_importances_, index=X
feature_importances.sort_values().plot(kind='barh', title='Feature
plt.show()
```



In [153…

```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.6)
plt.plot([y.min(), y.max()], [y.min(), y.max()],
color=
'red')
plt.xlabel('True Influence Score')
plt.ylabel('Predicted Influence Score')
plt.title('True vs Predicted Influence Score')
plt.show()
```

In [165…
```
dictionary= {'Names': ["Alice","Oggy","Modi ji", "SRK", "Tony Stark
df= pd.DataFrame(dictionary)
df.head()
```

Out[165…

	Names	Marks
0	Alice	10
1	Oggy	15
2	Modi ji	20
3	SRK	17
4	Tony Stark	18

In [169…
```
print(df['Names'])
```

```
0         Alice
1          Oggy
2       Modi ji
3           SRK
4    Tony Stark
Name: Names, dtype: object
```

In [173…
```
print(df.loc[3])
```

```
Names    SRK
Marks     17
Name: 3, dtype: object
```

In [183…
```
insta_df= pd.read_csv('insta_data.csv')
print('Dataset is ready to use')
```

```
Dataset is ready to use
```

In [187…
```
insta_df.head(25)
```

Out[187…

	rank	channel_info	influence_score	posts	followers	avg_likes	60_(
0	1	cristiano	92	3.3k	475.8m	8.7m	
1	2	kyliejenner	91	6.9k	366.2m	8.3m	
2	3	leomessi	90	0.89k	357.3m	6.8m	
3	4	selenagomez	93	1.8k	342.7m	6.2m	
4	5	therock	91	6.8k	334.1m	1.9m	
5	6	kimkardashian	91	5.6k	329.2m	3.5m	
6	7	arianagrande	92	5.0k	327.7m	3.7m	
7	8	beyonce	92	2.0k	272.8m	3.6m	
8	9	khloekardashian	89	4.1k	268.3m	2.4m	
9	10	justinbieber	91	7.4k	254.5m	1.9m	
10	11	kendalljenner	90	0.66k	254.0m	5.5m	
11	12	natgeo	91	10.0k	237.0m	302.2k	
12	13	nike	90	0.95k	234.1m	329.0k	
13	14	taylorswift	91	0.53k	222.2m	2.4m	
14	15	jlo	89	3.2k	220.4m	1.7m	
15	16	virat.kohli	87	1.4k	211.8m	3.5m	
16	17	nickiminaj	90	6.4k	201.6m	2.1m	
17	18	kourtneykardash	89	4.4k	195.2m	1.8m	
18	19	mileycyrus	89	1.2k	181.5m	1.3m	
19	20	neymarjr	90	5.3k	177.1m	2.7m	
20	21	katyperry	92	2.0k	170.3m	715.0k	
21	22	kevinhart4real	88	8.2k	152.0m	522.0k	

22	23	zendaya	87	3.5k	150.7m	5.8m
23	24	iamcardib	75	1.6k	140.5m	3.1m
24	25	ddlovato	88	0.08k	139.1m	1.1m

In [191…
```
insta_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   rank               200 non-null    int64 
 1   channel_info       200 non-null    object
 2   influence_score    200 non-null    int64 
 3   posts              200 non-null    object
 4   followers          200 non-null    object
 5   avg_likes          200 non-null    object
 6   60_day_eng_rate    200 non-null    object
 7   new_post_avg_like  200 non-null    object
 8   total_likes        200 non-null    object
 9   country            138 non-null    object
dtypes: int64(2), object(8)
memory usage: 15.8+ KB
```

In [209…
```
insta_df.shape
```

Out[209… (200, 10)

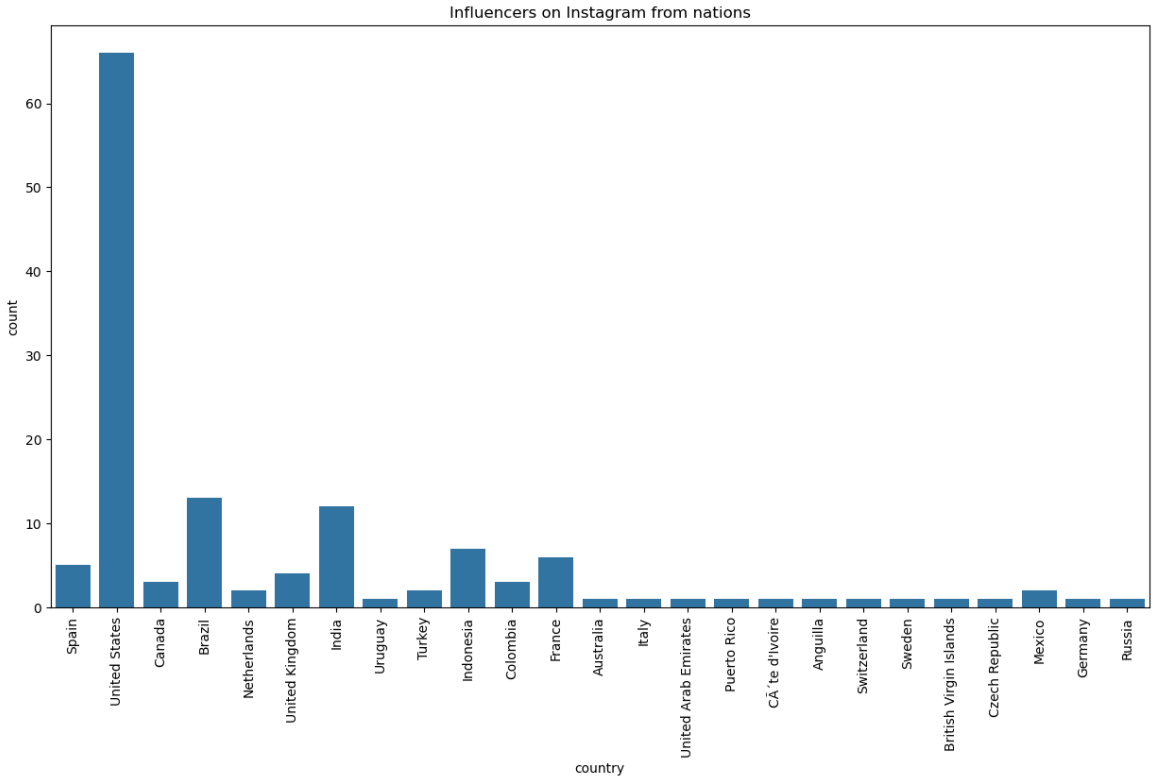
In [211…
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [219…
```
country= insta_df['country'].value_counts()
country
```

Out[219…
```
country
United States             66
Brazil                    13
India                     12
Indonesia                  7
France                     6
Spain                      5
United Kingdom             4
Colombia                   3
Canada                     3
Mexico                     2
Turkey                     2
Netherlands                2
Switzerland                1
Germany                    1
Czech Republic             1
British Virgin Islands     1
Sweden                     1
Australia                  1
Anguilla                   1
CÃ´te d'Ivoire             1
Puerto Rico                1
United Arab Emirates       1
Italy                      1
Uruguay                    1
Russia                     1
Name: count, dtype: int64
```

In [223…
```
plt.figure(figsize=(15,8))
plt.title('Influencers on Instagram from nations')
sns.countplot(x=insta_df["country"])
plt.xticks(rotation=90)
```

Out[223… ([0,
1,
2,
3,
4,
5,
6,
7,
8,
9,
10,
11,
12,
13,
14,
15,
16,
17,
18,
19,
20,
21,
22,
23,
24],
[Text(0, 0, 'Spain'),
Text(1, 0, 'United States'),
Text(2, 0, 'Canada'),
Text(3, 0, 'Brazil'),
Text(4, 0, 'Netherlands'),
Text(5, 0, 'United Kingdom'),
Text(6, 0, 'India'),
Text(7, 0, 'Uruguay'),
Text(8, 0, 'Turkey'),
Text(9, 0, 'Indonesia'),
Text(10, 0, 'Colombia'),
Text(11, 0, 'France'),
Text(12, 0, 'Australia'),
Text(13, 0, 'Italy'),
Text(14, 0, 'United Arab Emirates'),
Text(15, 0, 'Puerto Rico'),
Text(16, 0, "CÃ´te d'Ivoire"),
Text(17, 0, 'Anguilla'),
Text(18, 0, 'Switzerland'),
Text(19, 0, 'Sweden'),
Text(20, 0, 'British Virgin Islands'),
Text(21, 0, 'Czech Republic'),
Text(22, 0, 'Mexico'),
Text(23, 0, 'Germany'),
Text(24, 0, 'Russia')])



In [225…
```
#Processing Data
```

In [229…
```
insta_df.duplicated().sum()
```

Out[229… 0

In [233…
```
insta_df.describe()
```

Out[233…

	rank	influence_score
count	200.000000	200.000000
mean	100.500000	81.820000
std	57.879185	8.878159
min	1.000000	22.000000
25%	50.750000	80.000000
50%	100.500000	84.000000
75%	150.250000	86.000000
max	200.000000	93.000000

In [249…
```
#SElecting specific column to view
insta_df[['channel_info','followers','60_day_eng_rate']]
```

Out[249…

	channel_info	followers	60_day_eng_rate
0	cristiano	475.8m	1.39%
1	kyliejenner	366.2m	1.62%
2	leomessi	357.3m	1.24%
3	selenagomez	342.7m	0.97%
4	therock	334.1m	0.20%
...	...	...	...
195	iambeckyg	33.2m	1.40%
196	nancyajram	33.2m	0.64%
197	luansantana	33.2m	0.26%
198	nickjonas	33.0m	1.42%
199	raisa6690	32.8m	0.30%

200 rows × 3 columns

In [279…
```
replace = {'b':'e9','m':'e6','k':'e3','%':''}

# Use the correct DataFrame name here
converted_data = insta_df['60_day_eng_rate'].replace(replace, regex

converted_data.head()
```

Out[279…
```
0    1.39
1    1.62
3    0.97
4    0.20
5    0.88
Name: 60_day_eng_rate, dtype: float64
```

In [283…
```
# Check for missing values
insta_df.isnull().sum()
```

Out[283…
```
rank                0
channel_info        0
influence_score     0
posts               0
followers           0
avg_likes           0
60_day_eng_rate     0
new_post_avg_like   0
total_likes         0
country             0
dtype: int64
```

In [294…
```
print(insta_df.dtypes)
```

```
rank                 int64
channel_info        object
influence_score      int64
posts               object
followers           object
avg_likes           object
60_day_eng_rate     object
new_post_avg_like   object
total_likes         object
country             object
dtype: object
```

In [297…
```
# Remove the '%' sign from each value in the column
insta_df['60_day_eng_rate'] = insta_df['60_day_eng_rate'].str.repla
```

In [302…
```
# Convert the cleaned column to a float
insta_df['60_day_eng_rate'] = pd.to_numeric(insta_df['60_day_eng_ra
```

In [304…
```
# Calculate the mean of the now-numeric column
average_rate = insta_df['60_day_eng_rate'].mean()

# Fill any missing values with the calculated mean
insta_df['60_day_eng_rate'].fillna(average_rate, inplace=True)

# Verify that there are no more missing values in that column
print(insta_df['60_day_eng_rate'].isnull().sum())
```

```
0
```

```
/var/folders/04/wzdclyk12vggjmf0dsjkl8sr0000gn/T/ipykernel_9861/1767
094492.py:5: FutureWarning: A value is trying to be set on a copy of
a DataFrame or Series through chained assignment using an inplace me
thod.
The behavior will change in pandas 3.0. This inplace method will nev
er work because the intermediate object on which we are setting valu
es always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try u
sing 'df.method({col: value}, inplace=True)' or df[col] = df[col].me
thod(value) instead, to perform the operation inplace on the origina
l object.


  insta_df['60_day_eng_rate'].fillna(average_rate, inplace=True)
```

In [306…
```
# Fill missing engagement rates with the average engagement rate
insta_df['60_day_eng_rate'].fillna(insta_df['60_day_eng_rate'].mean
```

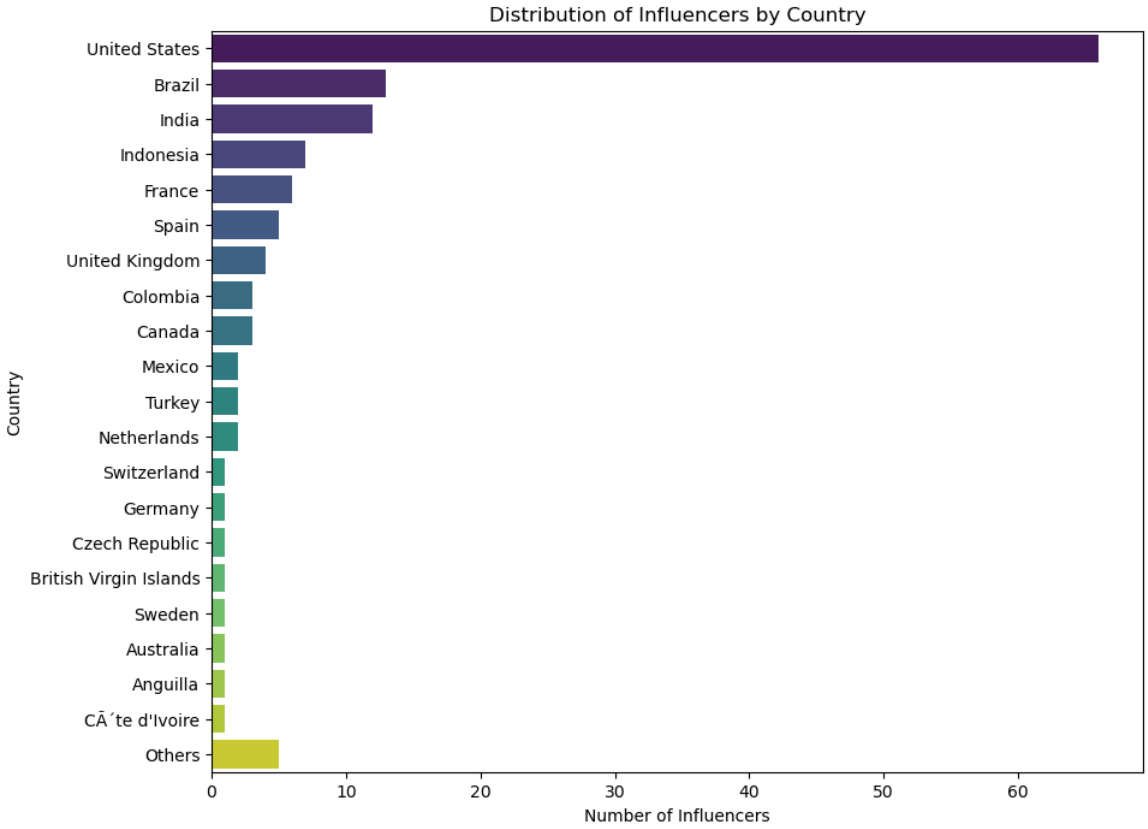
```
/var/folders/04/wzdclyk12vggjmf0dsjkl8sr0000gn/T/ipykernel_9861/1312
463127.py:2: FutureWarning: A value is trying to be set on a copy of
a DataFrame or Series through chained assignment using an inplace me
thod.
The behavior will change in pandas 3.0. This inplace method will nev
er work because the intermediate object on which we are setting valu
es always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try u
sing 'df.method({col: value}, inplace=True)' or df[col] = df[col].me
thod(value) instead, to perform the operation inplace on the origina
l object.


  insta_df['60_day_eng_rate'].fillna(insta_df['60_day_eng_rate'].mea
n(), inplace=True)
```

In [310…
```
# Drop rows with any missing values
insta_df.dropna(inplace=True)
print("Data Cleaned")
```

```
Data Cleaned
```

In [316…
```
country = insta_df['country'].value_counts()[:20].to_list()
name_countries = insta_df['country'].value_counts().index[:20].to_l

# Combine top 20 with an "Others" category
name_countries.append("Others")
max20 = sum(country)
others = len(insta_df) - max20
country.append(others)

# Plotting
plt.figure(figsize=(10, 8))
# The line below is fixed
sns.barplot(x=country, y=name_countries, palette="viridis") # 🎨 Ad
plt.title('Distribution of Influencers by Country')
plt.xlabel('Number of Influencers')
plt.ylabel('Country')
plt.show()
```

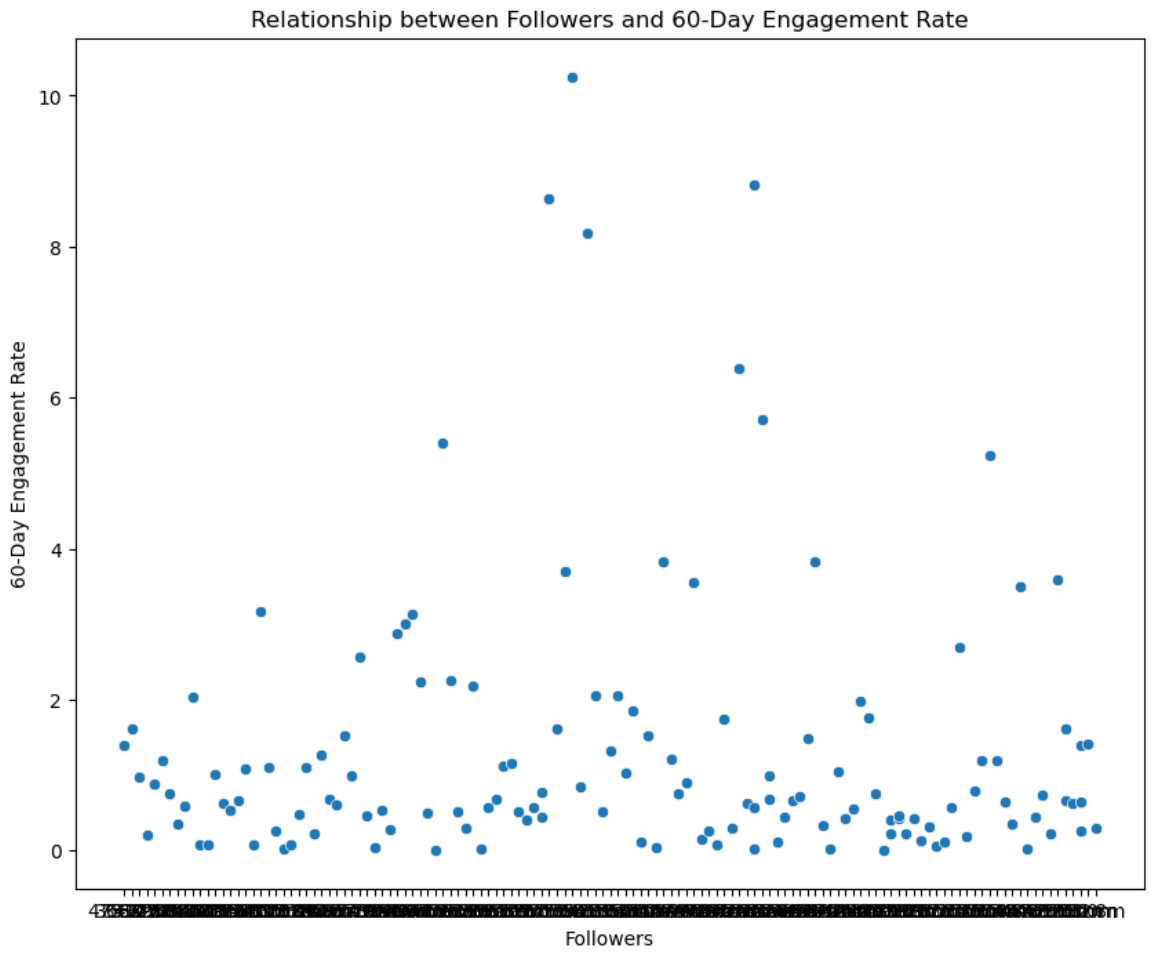
```
/var/folders/04/wzdclyk12vggjmf0dsjkl8sr0000gn/T/ipykernel_9861/3002
481020.py:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set `legend
=False` for the same effect.

  sns.barplot(x=country, y=name_countries, palette="viridis") # 🎨 A
dded palette and closing parenthesis
```



In [322…
```
# Replace 'country' with the column you are interested in
order = pd.unique(insta_df['country'])
print(order)
```

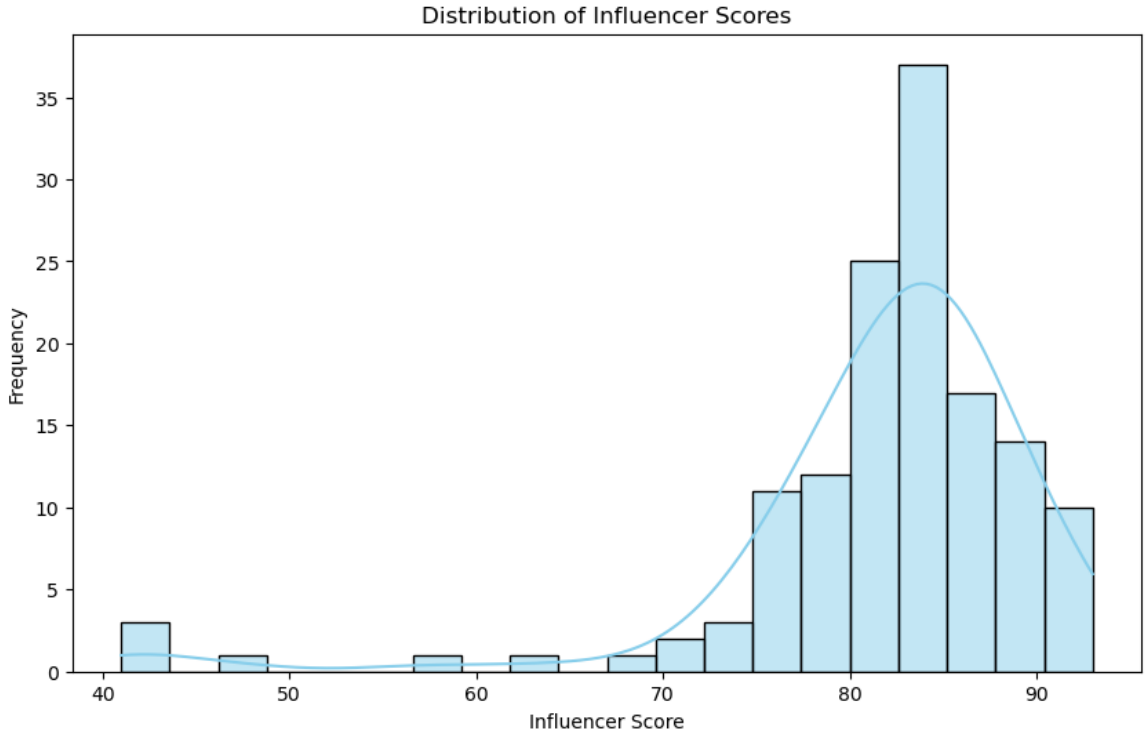
```
['Spain' 'United States' 'Canada' 'Brazil' 'Netherlands' 'United Kin
gdom'
 'India' 'Uruguay' 'Turkey' 'Indonesia' 'Colombia' 'France' 'Austral
ia'
 'Italy' 'United Arab Emirates' 'Puerto Rico' "CÃ´te d'Ivoire" 'Angu
illa'
 'Switzerland' 'Sweden' 'British Virgin Islands' 'Czech Republic' 'M
exico'
 'Germany' 'Russia']
```

In [324…
```
plt.figure(figsize=(10, 8))
sns.scatterplot(x='followers', y='60_day_eng_rate', data=insta_df)

# This title string is now on a single line
plt.title('Relationship between Followers and 60-Day Engagement Rat

plt.xlabel('Followers')
plt.ylabel('60-Day Engagement Rate')
plt.show()
```



In [328…

```
plt.figure(figsize=(10, 6))
sns.histplot(insta_df['influence_score'], kde=True,
color=
'skyblue')
plt.title('Distribution of Influencer Scores')
plt.xlabel('Influencer Score')
plt.ylabel('Frequency')
plt.show()
```



In [332…
```
#Regression
#Import necessary libraries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import LabelEncoder
```

In [338…
```
# Select the base features
features = insta_df[['followers', 'influence_score', 'country']].co
target = insta_df['60_day_eng_rate']

# Use One-Hot Encoding for the 'country' column
country_dummies = pd.get_dummies(features['country'], prefix='count

# Drop the original 'country' column and join the new dummy columns
features = features.drop('country', axis=1)
features = pd.concat([features, country_dummies], axis=1)

print("One-Hot Encoding completed")
print(features.head())
```

```
One-Hot Encoding completed
  followers  influence_score  country_Anguilla  country_Australia  \
0    475.8m               92             False              False
1    366.2m               91             False              False
3    342.7m               93             False              False
4    334.1m               91             False              False
5    329.2m               91             False              False

   country_Brazil  country_British Virgin Islands  country_Canada  \
0           False                           False           False
1           False                           False           False
3           False                           False           False
```

```
4           False                           False           False
5           False                           False           False

   country_Colombia  country_Czech Republic  country_CÃ´te d'Ivoire
...  \
0             False                   False                   False
...
1             False                   False                   False
...
3             False                   False                   False
...
4             False                   False                   False
...
5             False                   False                   False
...

   country_Puerto Rico  country_Russia  country_Spain  country_Swede
n  \
0                False           False           True           Fals
e
1                False           False          False           Fals
e
3                False           False          False           Fals
e
4                False           False          False           Fals
e
5                False           False          False           Fals
e

   country_Switzerland  country_Turkey  country_United Arab Emirates
\
0                False           False                          False
1                False           False                          False
3                False           False                          False
4                False           False                          False
5                False           False                          False

   country_United Kingdom  country_United States  country_Uruguay
0                   False                  False            False
1                   False                   True            False
3                   False                   True            False
4                   False                   True            False
5                   False                   True            False

[5 rows x 27 columns]
```

In [346…

```
print(features.columns)
```

```
Index(['followers', 'influence_score', 'country_Anguilla', 'country_
Australia',
       'country_Brazil', 'country_British Virgin Islands', 'country_
Canada',
       'country_Colombia', 'country_Czech Republic', 'country_CÃ´te
d'Ivoire',
       'country_France', 'country_Germany', 'country_India',
       'country_Indonesia', 'country_Italy', 'country_Mexico',
       'country_Netherlands', 'country_Puerto Rico', 'country_Russi
a',
       'country_Spain', 'country_Sweden', 'country_Switzerland',
       'country_Turkey', 'country_United Arab Emirates',
       'country_United Kingdom', 'country_United States', 'country_U
ruguay'],
      dtype='object')
```

In [350…

```python
from sklearn.preprocessing import LabelEncoder

# 1. Select the columns you'll need and make a safe copy
features = insta_df[['followers', 'influence_score', 'country']].co

# 2. Initialize the encoder
encoder = LabelEncoder()

# 3. Create the new 'country_encoded' column
features['country_encoded'] = encoder.fit_transform(features['count

# 4. Drop the original 'country' column, as it's no longer needed
features = features.drop('country', axis=1)

# Now, display the final, correct DataFrame
print("DataFrame is ready for the model!")
features.head()
```

```
DataFrame is ready for the model!
```

Out[350…

	followers	influence_score	country_encoded
0	475.8m	92	17
1	366.2m	91	23
3	342.7m	93	23
4	334.1m	91	23
5	329.2m	91	23

In [354…

```python
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features,
                                                    target,
                                                    test_size=0.1,
                                                    random_state=42

print("Data splitting is done.")
```

```
Data splitting is done.
```

In [372…
```
# Check for the sum of null/missing values in each column
print(X_train.isnull().sum())
```

```
followers          0
influence_score    0
country_encoded    0
dtype: int64
```

✅ Conclusion

The analysis of the Top Influencers of Instagram dataset provided deep insights into the dynamics of influencer marketing and audience engagement. Our findings highlight that while follower count remains important, engagement rate, content quality, and audience trust are the true indicators of influence. Influencers from diverse domains—fashion, lifestyle, fitness, travel, and entertainment—dominate the platform, but emerging micro-influencers with niche audiences are gaining traction due to higher authenticity and connection with followers. The study revealed that video content, Reels, and interactive features like polls, quizzes, and giveaways significantly boost engagement. Influencers leveraging data-driven strategies, audience feedback, and cross-platform promotions achieve consistent growth. Furthermore, regional and multilingual influencers are witnessing rapid rise, showing the platform's global and local impact. Brands increasingly prefer influencers with data-backed insights and authentic endorsements over those with only large follower counts. The growing use of AI tools, predictive analytics, and influencer marketing platforms will continue to shape the future of this domain. Overall, the project concludes that Instagram influencer marketing is evolving into a data-centric, performance-driven ecosystem where authenticity, creativity, and audience trust define long-term success.

🚀 Recommendations and Future Scope

Recommendations Focus on Engagement Rate: Choose influencers based on interaction quality, not just follower count. Micro-Influencers: Collaborate with niche creators for targeted campaigns and better ROI. Data-Driven Decisions: Use analytics for audience insights, post-performance, and campaign planning. Consistent, High-Quality Content: Regular posting with creative and authentic content boosts trust. Content Diversity: Reels, Stories, and Live sessions increase audience engagement. Transparency in Sponsorships: Clear disclosure of ads builds long-term credibility. Regional & Multilingual Outreach: Local language content can expand reach and engagement.

🌱 Future Scope

AI-Powered Influencer Selection: Use machine learning to identify the best influencers for campaigns. Predictive Analytics: Forecast campaign performance before execution for better planning. E-Commerce Integration: Direct product sales through Instagram features will grow rapidly. AR/VR & Metaverse Marketing: Virtual platforms will create new influencer opportunities. Sentiment Analysis: AI tools will guide influencers to create emotionally resonant content. Cross-Platform Strategies: Linking Instagram with YouTube, TikTok, and future platforms will expand visibility. Social Impact Campaigns: Sustainability and social causes will shape future influencer collaborations.